TECHNICAL REPORT NUMBER 3

THE REALIZATION OF TRANSFER FUNCTIONS WITH LOGICAL NETWORKS

Prepared By

DIGITAL SYSTEMS LABORATORY

C. C. CARROLL, PROJECT LEADER

August 1966

N67 19911

CONTRACT NAS8-20163

GEORGE C. MARSHALL SPACE FLIGHT CENTER

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

HUNTSVILLE, ALABAMA

APPROVED BY:

C H Holmes

C. H. Holmes
Head Professor
Electrical Engineering

SUBMITTED BY:

C C Carroll CHH

C. C. Carroll
Associate Professor of
Electrical Engineering

FOREWORD

This is a technical report of a study conducted by the Electrical Engineering Department of Auburn University under the auspices of the Auburn Research Foundation toward the fulfillment of the requirements prescribed in NASA Contract NAS8-20163.

ABSTRACT

In this work, three techniques for the synthesis of a network with a given transfer function are described. The synthesis techniques, which are based on approximation methods using sampled inputs, are listed and described as Methods I,II, and III. All three techniques are derived using the convolution integral, with each method originating from a different approximation of the convolution integral.

The purpose of this study is to determine the feasibility and the desirability of the digital implementation of a network with a given transfer function. The approach taken is entirely general and allows the techniques developed to be used in the realization of any transfer function. However, of the three techniques presented, the more advantageous technique will depend upon the specific problem under consideration.

In synthesizing a particular transfer function, the required amount of circuitry for the implementation of the network will determine the technique used. As will be noted, all three techniques require a varying amount of logic circuitry for the implementation of a specific network.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

There is a great deal of information available in the literature on the methods of realizing transfer functions. Network synthesis, a growing and already extremely large field, makes available many approaches and techniques for synthesizing a network with a tranfer function G(s). The synthesis of the network may be carried out in either the frequency or the time domain, depending upon whether the desired characteristics and available information are in the frequency domain or in the time domain. Synthesis in the frequency domain may start in the form of a required root-locus plot, or a graph of the required frequency response. Synthesis in the time domain may start with a graph of the desired transient response. However, these may be labeled as graphical or semi-graphical techniques; but, in some instances, an analytic approach might be preferred, in which case, analytic methods, such as the use of z transform theory or difference equations, are available.

In the techniques mentioned above it is assumed that the networks to be synthesized are analog in nature. As detailed in the following sections, an investigation was made of the use of digital networks in synthesizing a network with transfer functions G(s). The introduction to Chapter II gives a summary and discussion of analog and digital techniques that are presently available for network synthesis.

The remainder of Chapter II is devoted to the derivation and discussion of three new synthesis techniques. The techniques, referred to as Methods I, II, and III, are based on different approximations of the convolution integral. In deriving Method I, the input is assumed to be the sum of a number of step functions that closely approximate the actual input. In order to realize this assumption, the input is sampled at a frequency much higher than the highest frequency component of the input. In fact, all three techniques are based upon sampling the input and making certain valid assumptions. Method II employs the network response to a unit impulse response, while Method III is actually a rearrangement of the equations derived in Method I. Also included in Chapter II are illustrative examples to indicate the validity of the different methods.

Chapter III deals with the actual implementation of the logical networks, together with a discussion of the relative merits and the possibilities of extensions of each technique. These networks are presented in block diagram form.

II. SOME METHODS FOR THE SYNTHESIS OF TRANSFER FUNCTIONS

A. Introduction

Some of the methods presently available for the realization of transfer functions are presented in the Introduction. The results of these methods range from very good approximations to exact results, and are accomplished using either active, passive, or digital networks.

The network might be an RLC passive network which has the desired characteristics. On the other hand, active networks may be designed that have the characteristics of RLC networks, but without the use of an inductor. This was brought out in an investigation of compensating networks for an inertial guidance platform,[1] after which an operational amplifier was chosen as the compensating device. One of the advantages of this choice is that the entire compensating network can be microminiaturized.

Quite often compensation is accomplished using a digital system, called a digital controller, which is designed to perform certain linear operations on the input samples before delivering the output samples.[2] The digital controller may contain a passive network preceded and followed by synchronous samplers. The digital controller may also be a special purpose digital computer or a general purpose computer programmed to carry out the necessary operation.

Also, active linear digital controllers may be designed so that they accept a number sequence and process it to deliver a desired number sequence at their output.

Another method uses pulses to approximate the response of an analog network to a unit step input.[3] Also, another powerful method used in synthesizing transfer functions is based on a power series approximation of the system impulse response.[4] The following sections of this chapter deal with the methods I, II, and III as mentioned in Chapter I.

B. Method I - Approximation of Input with Step Functions

Given the transfer function G(s), it is desired to find the output $e_2(t)$ of the system for any input $e_1(t)$.

$$E_2(s) = G(s)\,E_1(s) = [G(s)/s]\,[s\,E_1(s)] \qquad (1)$$

Since multiplication in the s domain corresponds to convolution in the time domain, the time response $e_2(t)$ may be found by the convolution of the inverse Laplace transforms of G(s)/s and s $E_1(s)$.[5]

$$\mathcal{L}^{-1}\,[G(s)/s] = \mathcal{L}^{-1}\,[A(s)] = A(t) \qquad (2)$$

where A(t) is the response to a unit-step input and is called the indicial response.

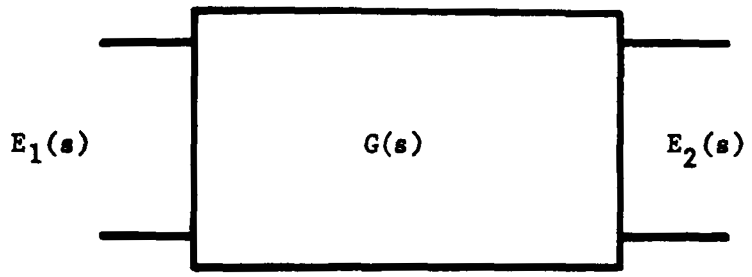


Fig. 1--Network with transfer function to be realized.

$$\mathcal{L}^{-1}\,[sE_1(s)] = \frac{de_1(t)}{dt} + e_1(0)\,\delta(t) \tag{3}$$

where $\delta(t)$ is an impulse function occurring at $t = 0$. Convoluting the results of equations (2) and (3) gives

$$e_2(t) = \left[\frac{de_1(t)}{dt} + e_1(0)\,\delta(t)\right] * A(t)$$

$$= \int_0^t \left[\frac{de_1(\tau)}{d\tau} + e_1(0)\,\delta(\tau)\right] A(t-\tau)\,d\tau,$$

and

$$e_2(t) = e_1(0)\,A(t) + \int_0^t \frac{de_1(\tau)}{d\tau}\,A(t-\tau)\,d\tau\,. \tag{4}$$

Equation (4) may be used to find the response $e_2(t)$ of a system for any input $e_1(t)$ provided the indicial response of the system is known.

Suppose $e_1(t)$ is piecewise continuous; i.e., it contains discontinuities. Equation (4) may still be used, but the effects of the discontinuities must be taken into account when finding the response of the system. Any discontinuity, either increasing or decreasing the value of $e_1(t)$, may be considered a step function applied at the time of the discontinuity $t = t_1$. The response of the system to the step function is $A(t - t_1)$ multiplied by an appropriate constant.

If the input is known as a function of time, equation (4) may be used to find the time response of the system. However, in general, the input is not known as a function of time, but is a measureable quantity. From this it is desired to build a logical network which will accept a general input and produce the same output as a network with transfer function G(s).

Since the input is a measureable quantity, let $e_1(t)$ be sampled at a frequency which is much higher than the highest frequency component of the input. Let the sampling frequency be f and the time between samples be T. The input may be approximated by letting $e_1(t)$ assume a constant value between sampling instants. This is illustrated in Figure 2, where it can be seen that, the higher the sampling frequency, the more accurate the approximation.

The integral in equation (4) may now be broken into parts and the response written as

$$e_2(t) = e_1(0)A(t) + \int_0^T \frac{de_1(\tau)}{d\tau} A(t-\tau)d\tau + \left[e_1(T)-e_1(0)\right]A(t-T)$$

$$+\int_T^{2T} \frac{de_1(\tau)}{d\tau} A(t-\tau) + \left[e_1(2T) - e_1(T)\right] A(t-2T) + \ldots$$

$$\int_{(n-1)T}^{nT} \frac{de_1(\tau)}{d\tau} A(t-\tau)d\tau + \left[e_1(nT)-e_1[(n-1)T]\right] A(t-nT).$$

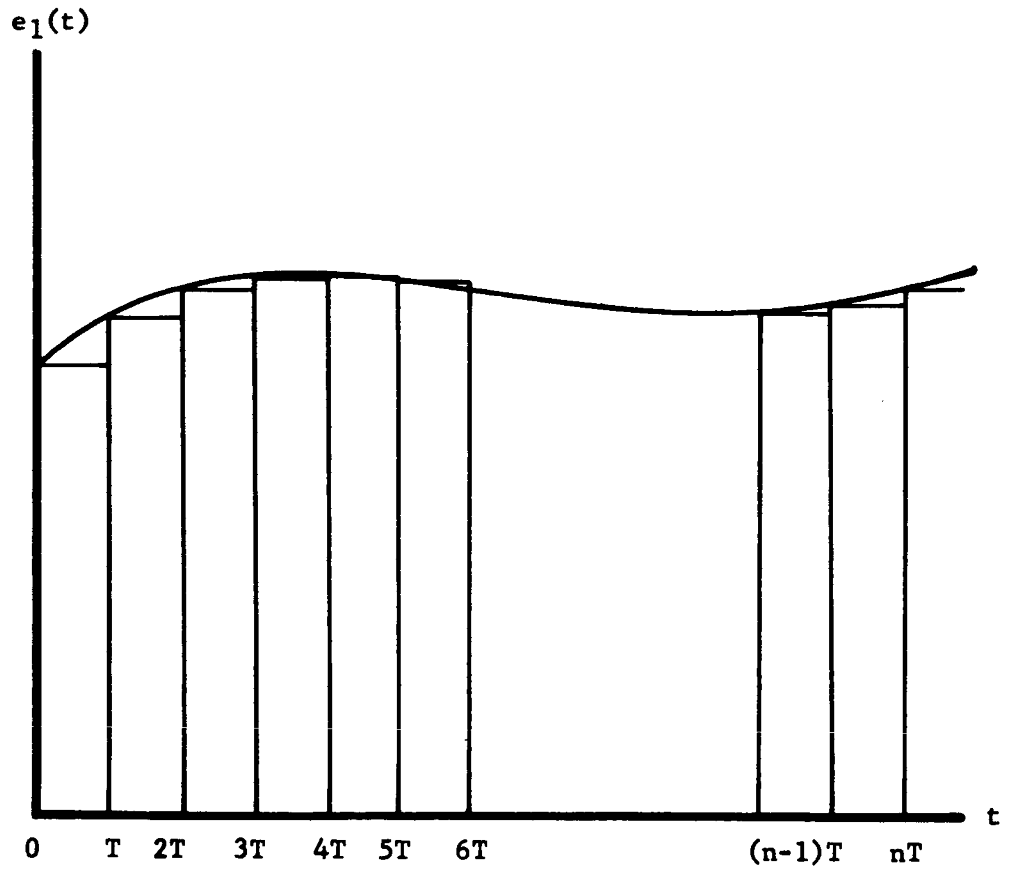


Fig. 2--Approximation of input with step functions.

However, since the input is assumed constant between sampling instants, the derivative inside the integrals is equal to zero and $e_2(t)$ becomes

$$\begin{aligned} e_2(t) = e_1(0)A(t) &+ \left[e_1(T) - e_1(0) \right] A(t-T) + \\ &+ \left[e_1(2T)-e_1(T) \right] A(t-2T) \\ &+ \ldots \left[e_1(nT)-e_1[(n-1)T] \right] A(t-nT), \end{aligned} \tag{5}$$

which results in the following closed form,

$$e_2(t) = \sum_{n=0}^{n=K} \left[e_1(nT)-e_1[(n-1)T] \right] A(t-nT) \tag{6}$$

where K denotes the K^{th} sample of the input. If the value of the response is needed at time $t = t_1$, where $KT \leq t_1 \leq (K + 1)T$ the summation is carried through $n = K$.

Equation (6) can also be derived by looking at Figure 2, which shows an approximation of $e_1(t)$ by considering the input to be the sum of a number of step functions occurring T seconds apart. The response of the system to a step function $u(t)$ is $A(t)$ and the response to a delayed step function $u(t-nT)$ is $A(t-nT)$. Therefore,

$$e_2(t) = e_1(0)A(t) + \left[e_1(T) - e_1(0)\right] A(t - T) + \ldots$$

$$+ \left[e_1(nT) - e_1[(n-1)T]\right] A(t-nT),$$

and

$$e_2(t) = \sum_{n=0}^{n=K} \left[e_1(nT) - e_1[(n-1)T]\right] A(t-nT),$$

which is the same as equation (6).

It should be noted that even though the input is sampled and is known at the sampling instant only, the output may be found at all instants of time using equation (6). Since the output is to be produced using logical networks, the output is needed at only the sampling instants and may be written

$$e_2(KT) = \sum_{n=0}^{n=K} \left[e_1(nT) - e_1[(n-1)T]\right] A[(K-n)T] \qquad (7)$$

where K denotes the sample and KT the time at which the response is needed. Let the term "characteristic constant" denote the specific values of A[(K-n)T] when evaluated properly using the appropriate K and n.

C. Method II - Use of Network Impulse Response

Again, consider the network with input, transfer function and output $E_1(s)$, $G(s)$ and $E_2(s)$ respectively:

$$E_2(s) = E_1(s)\, G(s)$$

$$\mathcal{L}^{-1}[G(s)] = g(t)$$

$$\mathcal{L}^{-1}[E_1(s)] = e_1(t).$$

Convoluting $g(t)$ and $e_1(t)$ gives

$$e_2(t) = e_1(t) * g(t) = \int_0^t e_1(\tau)\, g(t-\tau)\, d\tau \qquad (8)$$

where $g(t)$ is the network response to a unit impulse function.

Equation (8) allows the response $e_2(t)$ to be seen as the area under the curve which results when the two curves $e_1(\tau)$ and $g(t-\tau)$ are multiplied together. Using this property of the convolution integral, assume that the sampling frequency is such that the input may be considered constant between sampling instants. Equation (8) may then be broken into parts:

$$e_2(t) = e_1(0)\int_0^T g(t-\tau)\,d\tau + e_1(T)\int_T^{2T} g(t-\tau)\,d\tau + \ldots$$
$$\ldots + e_1(nT)\int_{nT}^{(n+1)T} g(t-\tau)\,d\tau, \text{ and} \qquad (9)$$

$$e_2(t) = \sum_{n=0}^{n=[t/T]} e_1(nT) \int_{nT}^{(n+1)T} g(t-\tau)\,d\tau \qquad (10)$$

where [t/T] denotes the greatest integer of t/T.

The response at the sampling instants is

$$e_2(KT) = \sum_{n=0}^{n=K} e_1(nT) \int_{nT}^{(n+1)T} g(KT - \tau)d\tau. \tag{11}$$

Thus, upon selection of a particular transfer function G(s), g(t) may be computed and inserted into equation (11) to find the response of the system to any input $e_1(t)$.

D. Method III - A Reinvestigation of Method I

If Method I is investigated further, a new and perhaps more advantageous approach can be made available. In deriving Method I, the input is approximated by a number of step functions with the output $e_2(t)$ being equal to the sum of the responses to these step functions.

The response may be written as before in equation (5):

$$e_2(t) = e_1(0)\, A(t) + \left[e_1(t) - e_1(0)\right] A(t-T)$$
$$+ \left[e_1(2T)-e_1(T)\right] A(t-2T) + \ldots + \left[e_1(nT)-e_1[(n-1)T]\right] A(t-nT).$$

Now, suppose the response is needed only at the sampling instants; i.e., at t = nT; then

$$e_2(nT) = e_1(0)\, A(nT) + \left[e_1(T) - e_1(0)\right] A[(n-1)T] +$$
$$\left[e_1(2T) - e_1(T)\right] A[(n-2)T] + \ldots + \left[e_1(nT)-e_1[(n-1)T]\right] A(0). \tag{12}$$

After multiplying term by term and regrouping the resulting terms, $e_2(nT)$ may be written as

$$e_2(nT) = e_1(0)\Big[A(nT) - A[(n-1)T]\Big] + e_1(T)\Big[A[(n-1)T] - A[(n-2)T]\Big] + \ldots + e_1(nT)\,A(0), \tag{13}$$

and

$$e_2(KT) = \sum_{n=0}^{n=K} e_1(nT)\Big[A[(K-n)T] - A[(K-n-1)T]\Big]. \tag{14}$$

This can be seen to have a definite advantage over Methods I and II insofar as implementation is concerned. However, it is important to note that this result can be obtained through the proper integration of the integral representation in Method II.

Thus, three different approaches in approximating the network response to a general input have yielded a common result, since the response found using Method III is equivalent to that of Method I and Method II.

E. Illustrative Examples

In order to illustrate the validity of the three techniques, suppose a network with transfer function is chosen as shown in Figure 3.

Method I is first used to find the response $e_2(KT)$ for several example inputs; this should be sufficient to illustrate the validity of all three techniques since the responses at the sampling instants are equivalent.

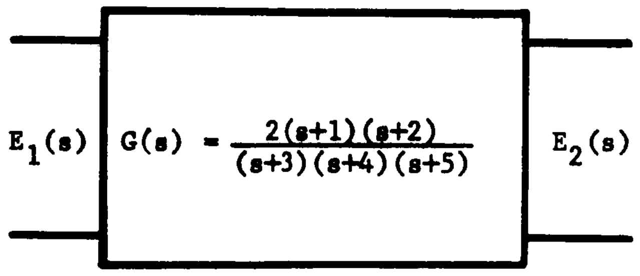


Fig. 3--Example system.

To find the indicial or step response let $e_1(t) = u(t)$.

$$E_1(s) = 1/s$$

$$E_2(s) = A(s) = E_1(s)\ G(s) = \frac{2(s+1)(s+2)}{s(s+3)(s+4)(s+5)}$$

$$A(t) = \mathcal{L}^{-1}\left[\frac{2(s+1)(s+2)}{s(s+3)(s+4)(s+5)}\right]$$

$$= \frac{1}{15} - \frac{2}{3} e^{-3t} + 3e^{-4t} - \frac{12}{5} e^{-5t} .$$

According to equation (6),

$$e_2(KT) = \sum_{n=0}^{n=K} \Big[e_1(nT) - e_1[(n-1)T]\Big] A[(K-n)T]. \qquad (6)$$

The response at time t = KT may be found by summing the difference in consecutive samples multiplied by the appropriate characteristic constants. Suppose the example inputs are chosen as follows:

(a) step input - $e_1(t) = u(t)$

(b) ramp input - $e_1(t) = t$

(c) parabolic input - $e_1(t) = t^2$

(d) sinusoidal input - $e_1(t) = 2 \sin t + 3 \sin 2t$

For the step input the difference in consecutive samples after the "n=0" sample is equal to zero, and $e_2(KT) = e_1(0)\ A[(K-n)T]$.

Thus the response found using Method I is exact at the sampling instants as shown in Figure 4.

Figure 5 shows the response of the network when the input is a ramp function, while Figure 6 gives the response to a parabolic input. The sinusoidal input was chosen because of its generality as an input function, and because its wave shape has both positive and negative slopes; the latter characteristic permits a better test for the approximations used in deriving the equations. The response to a sinusoidal input is shown in Figure 7.

These responses were found using Method I, but as stated before, Methods I, II and III give equivalent results when the response is found at the sampling instants only.

The sampling interval for the examples in Figures 4, 5, 6 and 7 is $T = 0.1$ seconds. In order to illustrate the effect of the sampling frequency on the accuracy of the response, let the sampling interval be halved, $T = 0.05$ seconds, and compute the response to the sinusoidal input. These results are shown in Figure 8. A comparison of the results in Figures 7 and 8 will readily show that the response found by using the faster sampling frequency is the more accurate. Hence, it should be obvious that a sampling frequency could be chosen that would permit the response to be computed as accurately as desired. It is this characteristic of the synthesis techniques which increases their adaptability to the digital computer.

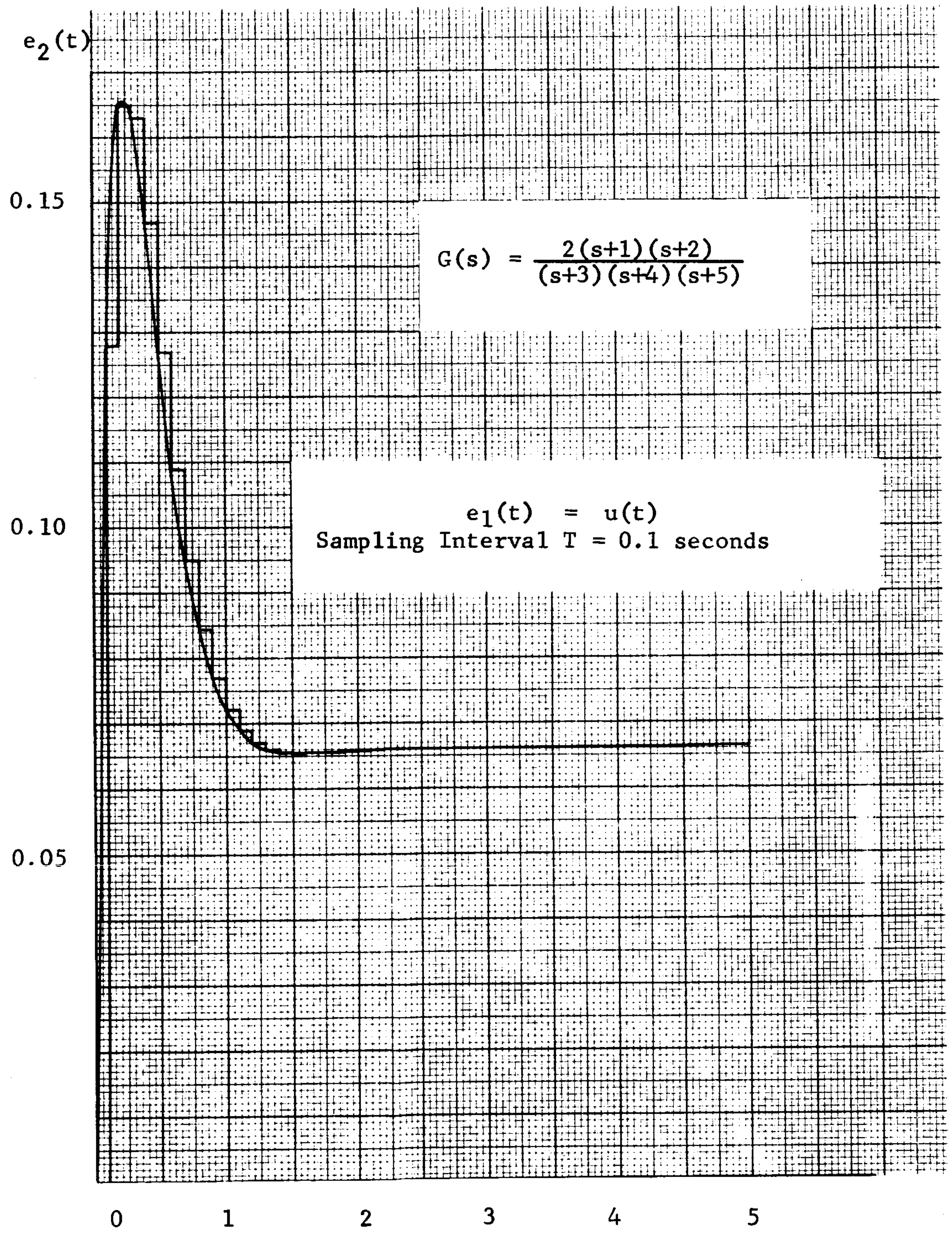


Fig. 4--Unit step response.

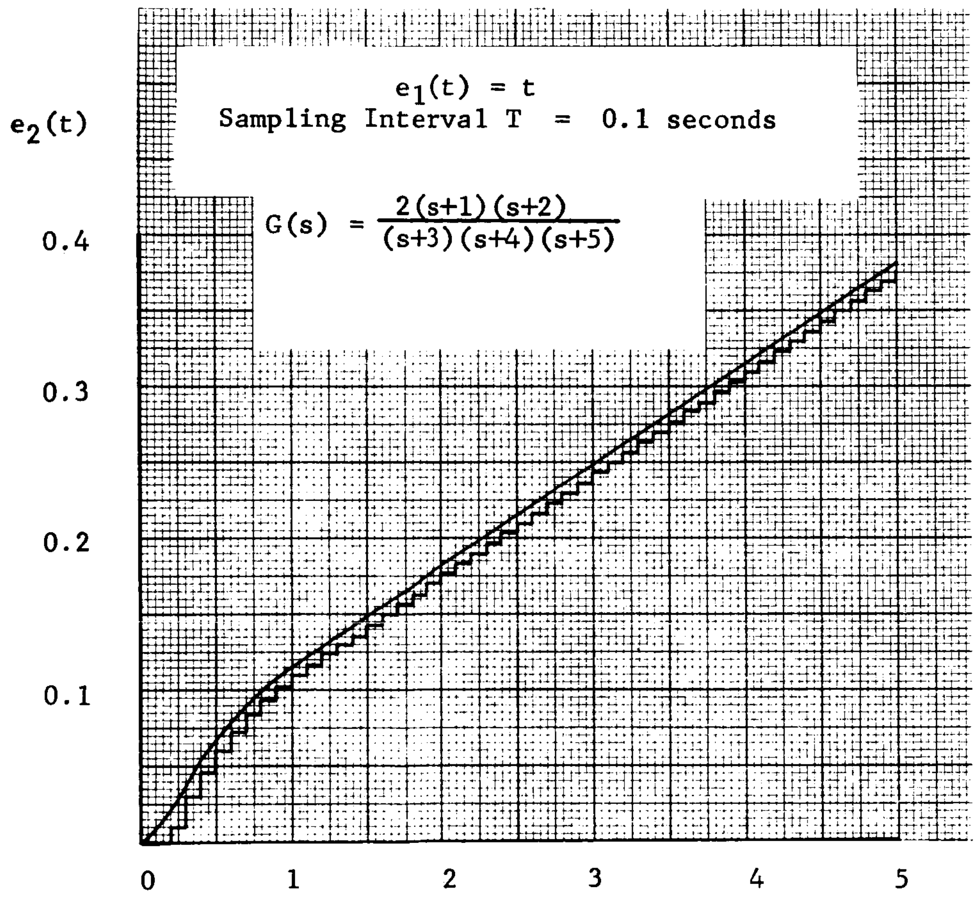


Fig. 5--Ramp response.

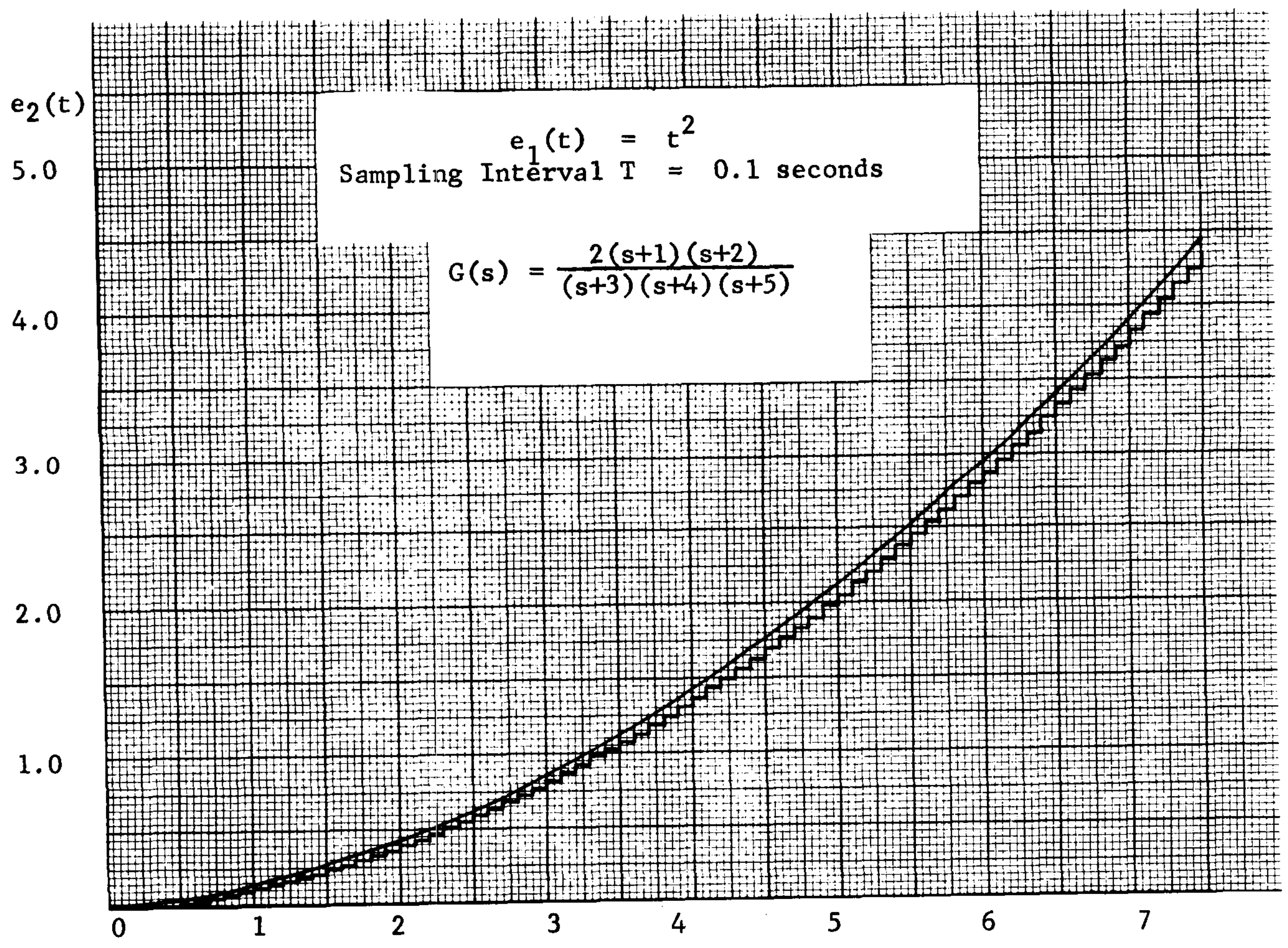


Fig. 6--Parabolic response.

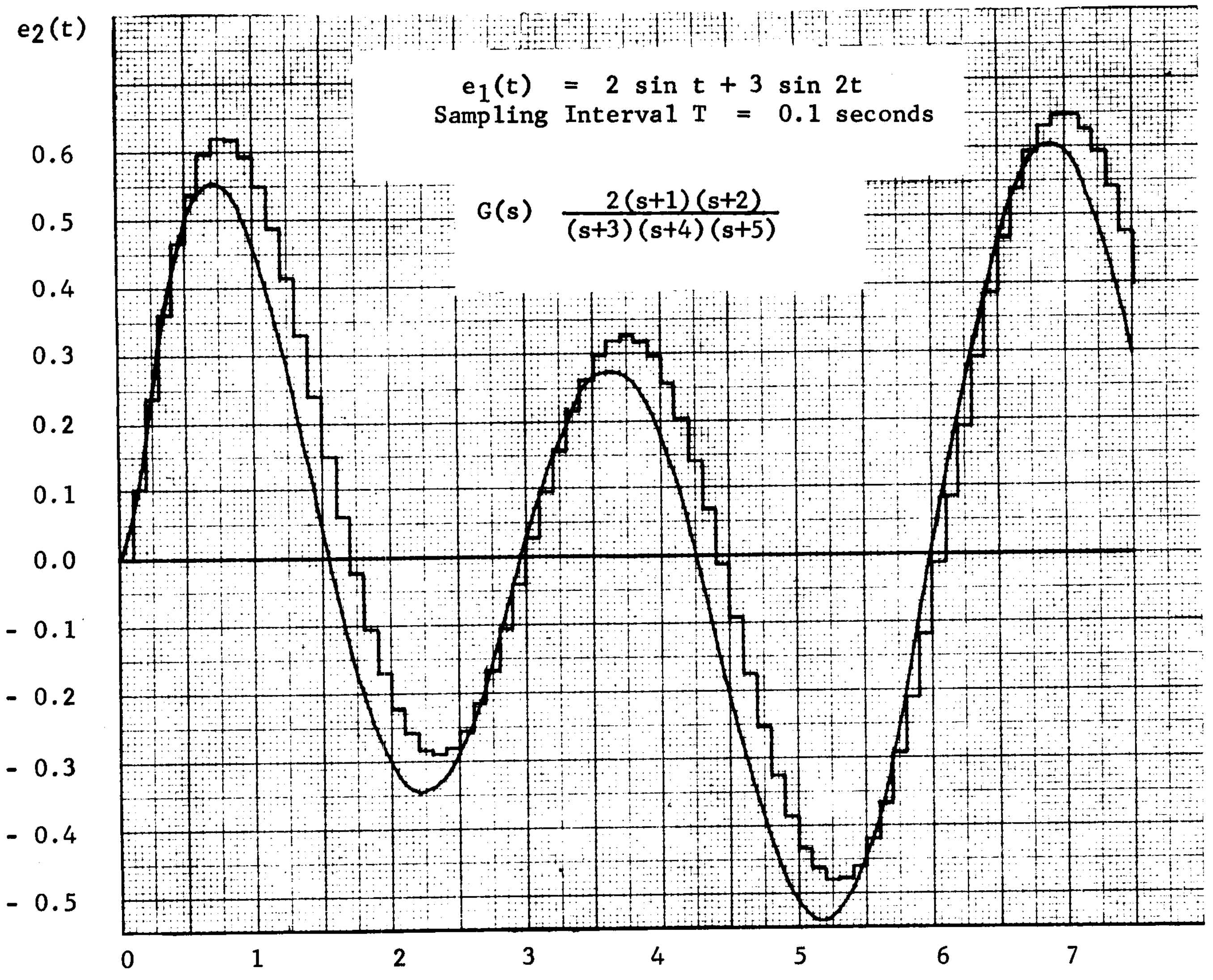


Fig. 7--Sinusoidal response.

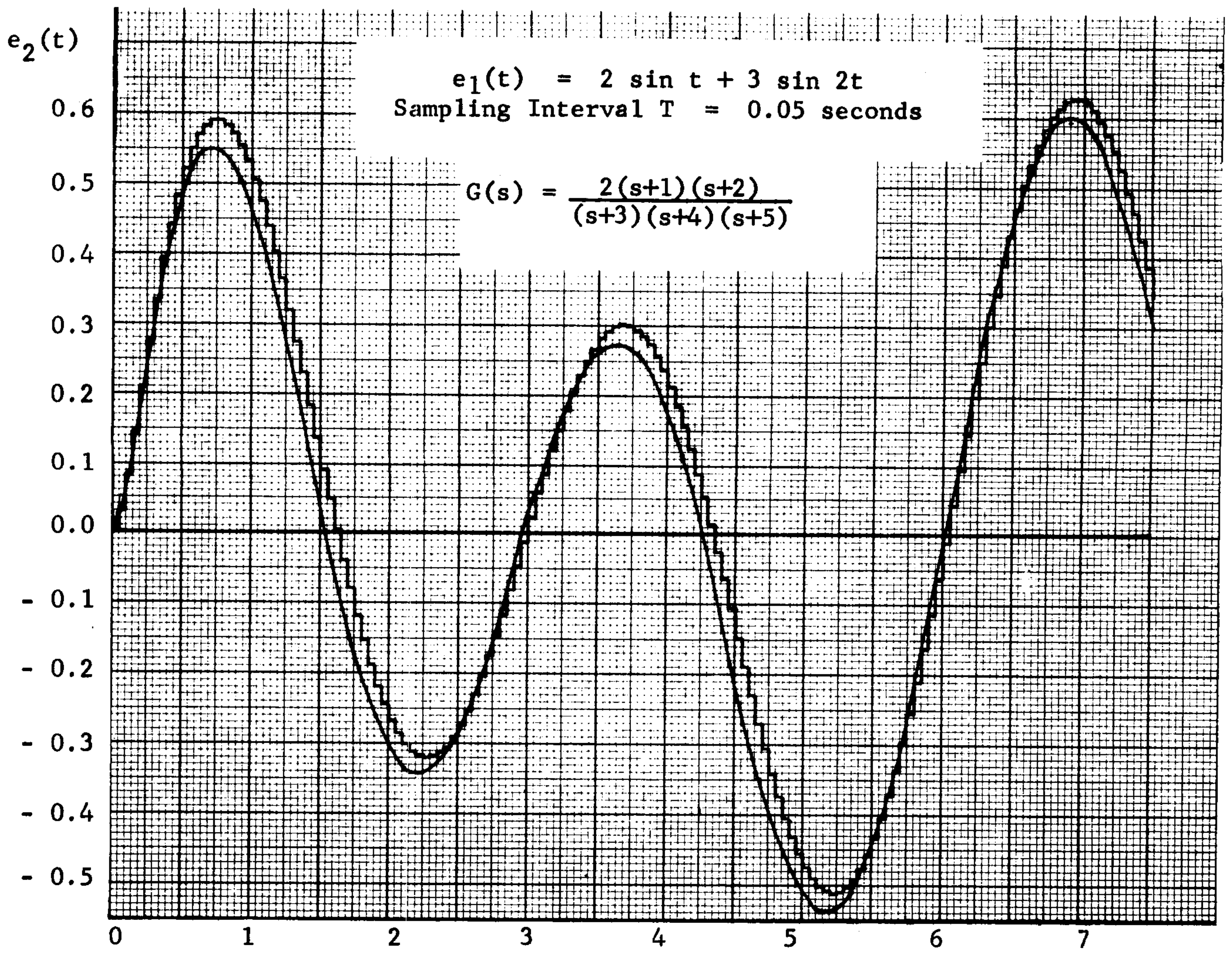


Fig. 8--Sinusoidal response with sampling interval halved.

III. IMPLEMENTATION USING METHODS I, II, AND III

A. Introduction

A transfer function G(s) may be realized by the implementation of one of the derived network equations. It is now important to discuss the actual implementation of the network response equations. First, there is the classical approach, wherein the logical network is comprised of an analog-to-digital converter, combinational circuitry (including shift registers for internal information storage), and a digital-to-analog converter.[6,7] For a second approach, the combinational circuitry is replaced by arithmetic units, shift registers, counters and gating circuitry. This approach offers possibilities for a reduction in the complexity of the logical network as compared with the classical approach. It can be seen that equation (14), which is the basic equation of Method III, is the most practical of the network equations to implement. This will be discussed further in a later section devoted to the evaluation of each type of implementation and its possibilities for extension.

B. Method I

For convenience, rewrite equation (7):

$$e_3(KT) = \sum_{n=0}^{n=K} \left[e_1(nT) - e_1[(n-1)T] \right] A[(K-n)T]. \qquad (7)$$

It can be seen that the logical network necessary to implement this equation must contain circuitry for addition, subtraction and multiplication. This network is given in block diagram form in Figure 9.

A list of the logic function definitions is given in the following table:

w_1, w_2, w_3	input at the n^{th} sampling instant
x_1, x_2, x_3	input at the $(n-1)^{th}$ sampling instant
y_{i1}, y_{i2}, y_{i3}	i^{th} difference in consecutive samples
v_{i1}, v_{i2}, v_{i3}	output of i^{th} multiplier network
z_1, z_2, z_3, z_4	output of summing network

Once a transfer function is specified, A(t) can be found, and the values A[(k-n)T] can be computed and incorporated directly into the network as constant multipliers for the appropriate differences in the consecutive samples that are stored in shift register elements s_o, s_1, . . . , etc. The logical subtractor, multiplier and adder networks are simply combinational networks that perform the indicated operations.

From the circuit in Figure 9, it is evident that the required number of shift register elements and logical multipliers is determined by the sampling period T and the maximum length of time for which a response might be needed. This results from equation (7), requiring that each difference in consecutive samples be stored and multiplied at each sampling instant by a different characteristic constant.

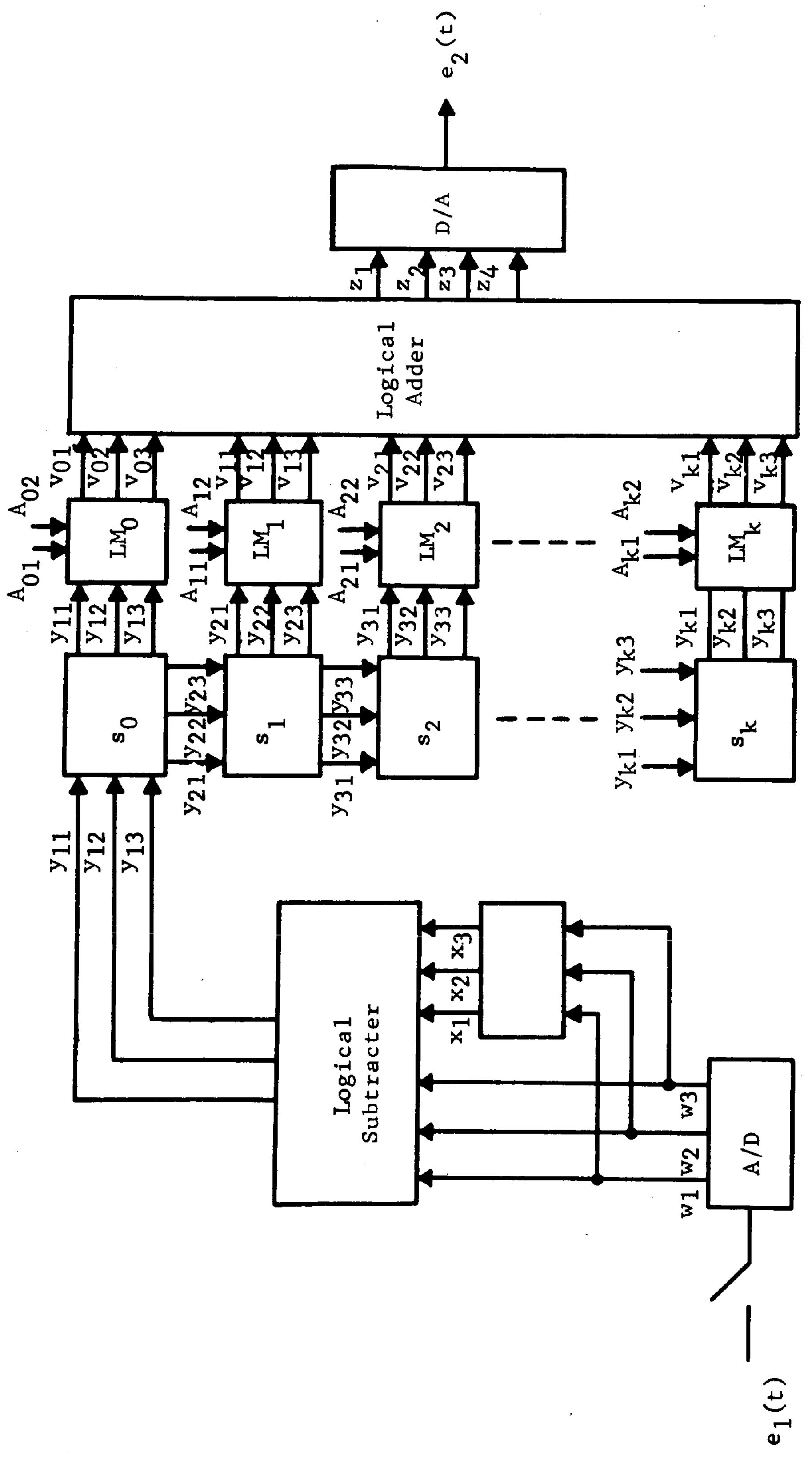


Fig. 9--Block diagram of implementation using Method I.

Therefore, for this type of implementation to be practical, A(t), the step response of the system with transfer function G(s), must be a decaying exponential response with a time constant of the order of the sampling period. This allows the logical network to be built with a finite and practical number of shift registers and logical multipliers. Suppose there are four required multiplier networks in the realization of a specific transfer function, and that the output of each is represented by three bits of digital information. In such a case, there are twelve inputs to the logical adder network, and, as will be seen later, this is not a small logical network. Therefore, it is desirable to devise methods other than combinational circuitry to serve as an adder network.

Equation (7), which yields the response at the sampling instants, and equation (6), which yields the continuous response, may also be implemented with a digital computer. This can be especially useful in time-sharing systems that use on-line digital computers. Equations (6) and (7) have been programmed on a digital computer, and both have yielded highly accurate results.

C. Method II and Method III

As stated earlier, after proper integration of the integral representation in equation (14), the results, and, therefore, the implementation, are identical in Methods II and III. The basic network equation of Method III is equation (14):

$$e_2(KT) = \sum_{n=0}^{n=K} e_1(nT) \left[A[(K-n)T] - A[(K-n-1)T]\right] \qquad (14)$$

The logical network used to implement equation (14) is shown in block diagram form in Figure 10. Equation (14) does not require the difference in consecutive samples, but does require the difference in consecutive time values of the characteristic response A(t). These characteristic constants may be computed and incorporated directly into the logical network as constant multipliers for the appropriate input samples. It should be noted that this type of implementation can be used to approximate the response of a system at the sampling instants by terminating the series obtained from equation (14) after a certain number of input samples. The step response of a general, stable system approaches a constant value after a finite length of time, and, therefore, the difference in consecutive time values of this characteristic response approaches zero. This allows the termination of the series in equation (14), which corresponds in the logical network to omitting those multiplier circuits whose characteristic constant inputs are negligible and will produce a negligible output as far as the summing network is concerned. There follows an example of a network whose characteristic constants are negligible after four consecutive time constants; i.e., A(4T) - A(3T), A(5T) - A(4T), . . . etc., are approximately equal to zero.

After computing the values for A[(K-n)T] - A[(K-n-1)T] it is necessary to select the number of bits required to represent A[(K-n)T] - A[(K-n-1)T] and $e_1(nT)$ correctly. In order to keep the logical network

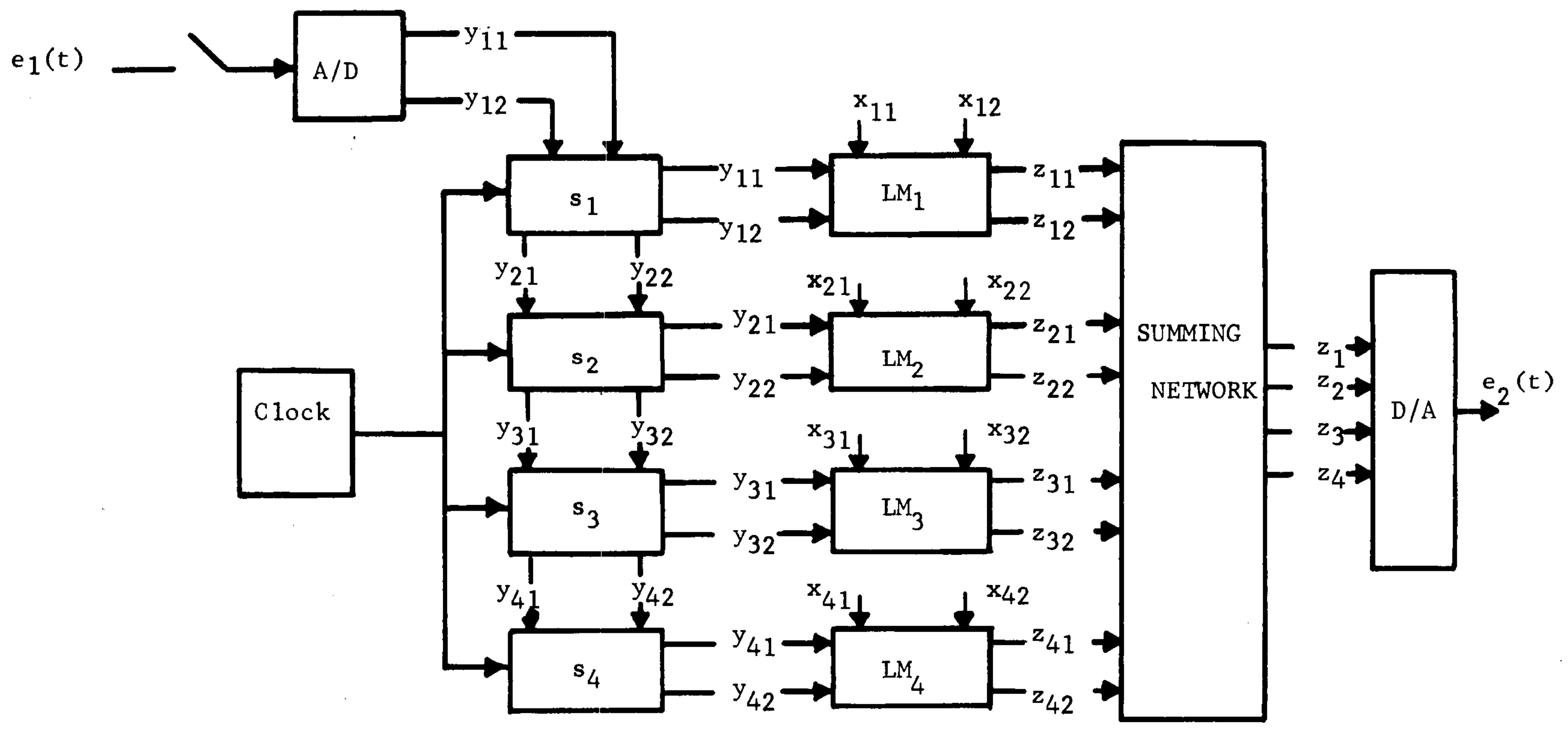


Fig. 10--Block diagram of implementation using Method II and Method III.

as simple as possible while, at the same time, representing the functions accurately, two bit precision was chosen for the input and multiplier networks; the output of the summing network is represented with four bits. A list of the logic function definitions is given in the table below:

y_{i1}, y_{i2}	sampled input to the ith shift register
x_{i1}, x_{i2}	A[(K-n)T] - A[(K-n-1)T] input to the ith multiplier network
z_{i1}, z_{i2}	output of the ith multiplier network
z_1, z_2, z_3, z_4	output of the summing network

MULTIPLIER NETWORK OUTPUT FUNCTIONS

A_i		$e_i(nT)$		$f_i = A_i e_i(nT)$	
x_{11}	x_{12}	y_{11}	y_{12}	z_{11}	z_{12}
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

$z_{11} = x_{11}\, x_{12}\, y_{11}\, y_{12}$

$z_{12} = x_{11}\, x_{12}\, y_{11}'\, y_{12} + x_{11}\, x_{12}\, y_{11}\, y_{12}'$

x_{21}	x_{22}	y_{21}	y_{22}	z_{21}	z_{22}
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	0	1

$z_{21} = 0$

$z_{22} = x_1 x_2' y_1$

$x_{31}x_{32}$		$y_{31}y_{32}$		$z_{31}z_{32}$	
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	1

$z_{31} = 0$

$z_{32} = x_{31}' x_{32} y_{31} y_{32}$

However since $x_{41} = x_{31}$, $x_{42} = x_{32}$, $y_{41} = y_{31}$, and $y_{42} = y_{32}$,

$z_{41} = z_{31} = 0$

and $z_{42} = z_{32} = x_{31}' x_{32} y_{31} y_{32}$

The minimized logic functions for the output of the summing network are given below.

$z_1 = ACEG$

$z_2 = AEG' + ACG' + AC'G + CE'G + A'E'G + CEGH + BDE$
$+ BEH + DEH + BDG + BFG + DFG + BCF + AC'FH$
$+ CFH + BDFH + ADF + B'DF + ADH$

$$z_3 = ACEG' + AC'EG + A'CEG + ACE'G + BDE'G'F' + BCEH$$
$$+ ADEH + A'B'C'D'EG' + A'B'DEG'H' + BC'D'E'G'H'$$
$$+ BDEG + A'B'C'E'F'G + BC'D'E'GF' + AC'E'F'G'H'$$
$$+ A'CE'F'G'H' + A'DE'F'H + BC'D'E'H + ACFH + A'B'C'FH$$
$$+ A'B'CE'F'H + AC'DE'F'H + A'DFG'H' + ADFG + A'B'C'D'FG$$
$$+ BCFG + AC'D'FG'H' + A'BC'FG'H + A'B'CFG'H$$

$$z_4 = BDFH' + B'D'FH' + BD'FH + BD'FH + B'DFH + BDF'H$$
$$+ B'D'F'H + BD'F'H' + B'DF'H'$$

Where, for simplicity in writing the logic functions,

$A = z_{11}$, $B = z_{12}$, $C = z_{21}$, $D = z_{22}$, $E = z_{31}$, $F = z_{32}$, $G = z_{41}$, and $H = z_{42}$.

The summing network output functions z_1, z_2, z_3, and z_4 as shown, are not simple logic functions. Furthermore, it should be noted that, in this specific example, there is no provision for the polarity of the input samples, or for the output values of the response. In order to have a general system, a sign bit must be included with the actual information bits. This would tend to increase the complexity of the summing network output functions. Again, it is important to note that, in order for this type of implementation to be practical, a different network must be devised with which to sum the outputs of the multiplier networks.

Suppose the outputs of the multiplier networks are represented with a sign bit and two information bits. In an effort to decrease the complexity of the summing network, let the network shown in Figure 11 be used as described in the following paragraphs.

Let the outputs of the multiplier networks be binary weighted with the sign bit followed by the least to the most significant information bit. These outputs are used as inputs to a bank of shift registers, which will accept information inputs in parallel and will transfer this information out serially. As this information is transferred out it is "ANDED" with a pulse train from the clock. This produces a pulse train used as the input to an up-down counter that counts the number of pulses, thereby summing the outputs of the multiplier networks. This type of summing network is actually an accumulator that accepts the information from one section of the shift register band and either counts up or down, depending upon the sign bit preceding the information bits.

Suppose there are ten multiplier networks, with each network having two information bits plus a sign bit. This gives each network a range of minus three to plus three, and hence requires the up-down counter, to be able to produce an output of minus thirty to plus thirty. The up-down counter, however, has no provision for the sign of its output, and it is, therefore, preset at the binary value thirty before receiving information from the bank of shift registers. After all the information has been transferred out of the shift registers, the counter is again set to its null value of thirty. This allows all states above the null value to indicate positive outputs, and those below the null value to indicate negative outputs.

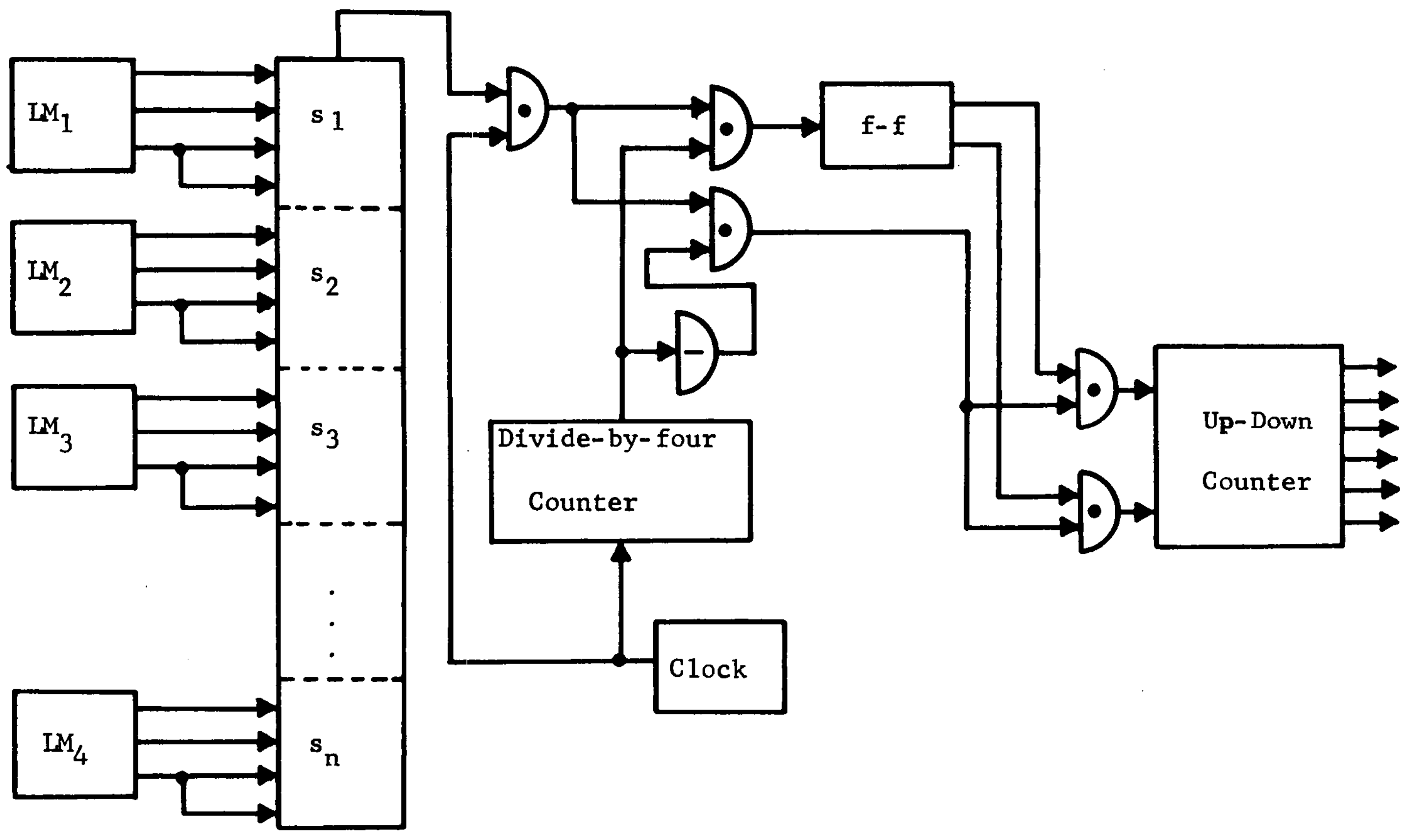


Fig. 11--Block diagram of summing network.

The up-down counter can actually be viewed as a sequential machine that accepts serial inputs and processes these inputs to produce an output which is a function of the inputs and the previous output. Figure 12 gives the timing diagrams for the divide-by-four counter.[8] This circuit simply allows every fourth pulse to pass, and this pulse is ANDED with the output pulse train from the shift register determining the sign of the information pulses to follow. An extension of this counter is used to count every thirtieth clock pulse that sets the up-down counter to the desired null level.

The logical gates and symbols in Figure 11 are illustrated in the appendix.

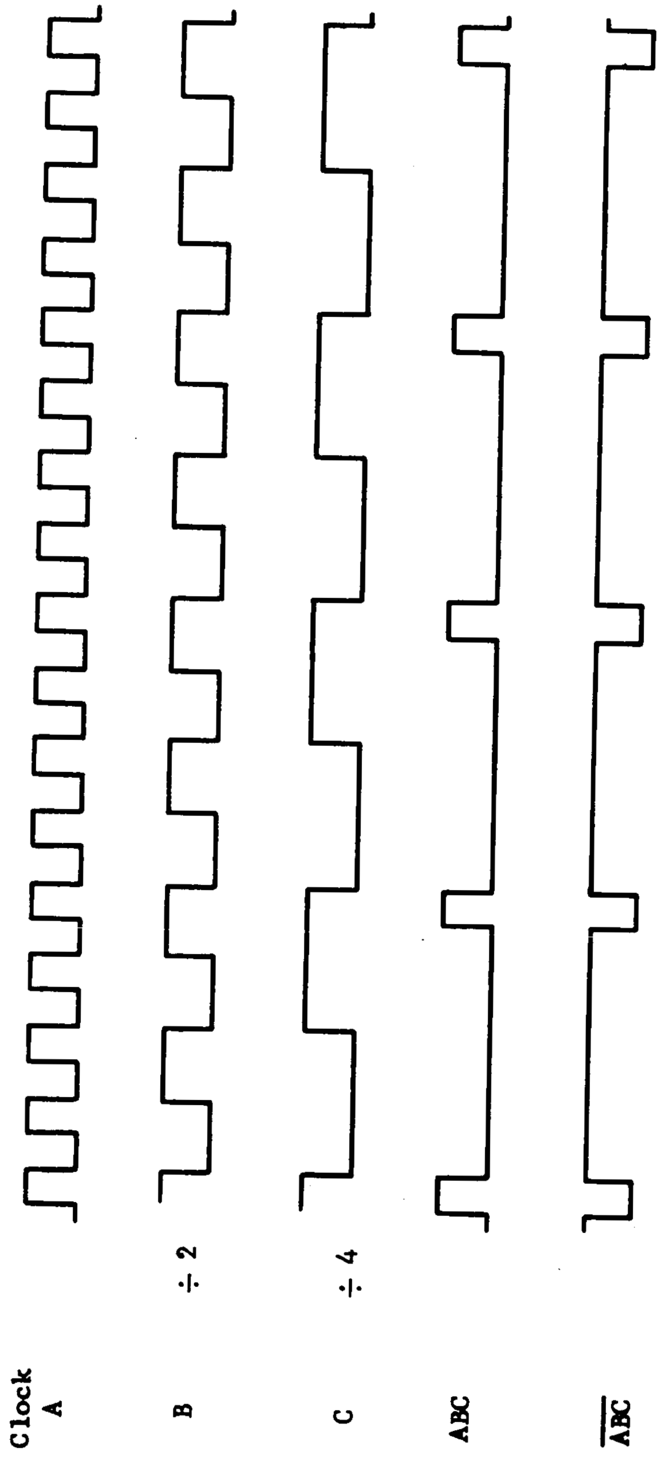


Fig. 12--Timing diagrams of the divide-by-four counter.

IV. A DIGITAL LEAD COMPENSATOR FOR AN ERROR CORRECTING CONTROL SYSTEM

Proportional plus derivative, or "lead" compensation, is often added to a single loop automatic control system in order to obtain a satisfactory response. This compensation may be accomplished with either analog or digital networks. An error correcting control loop with digital compensation in the feedback path is shown in Figure 13.

The unit step response of an analog lead compensator is given in Figure 14. The lead compensation is produced by summing the input and the derivative of the input. An approximation of this response can be obtained with a digital lead compensator,[3] as shown in Figure 15. The digital compensator consists of a monostable multivibrator, which produces the compensation pulse, a proportional type network, and a summing network. The compensation pulse is of width γ, the desired time of compensation, and amplitude α, which depends upon the magnitude of the input step.

A more accurate approximation of the analog network response can be obtained with a digital network that produces an output similar to the time response in Figure 14. In order to achieve this, the network must be of two sections: the first section produces a proportional output in digital form; and the other section produces an output sequence that represents the derivative compensation.

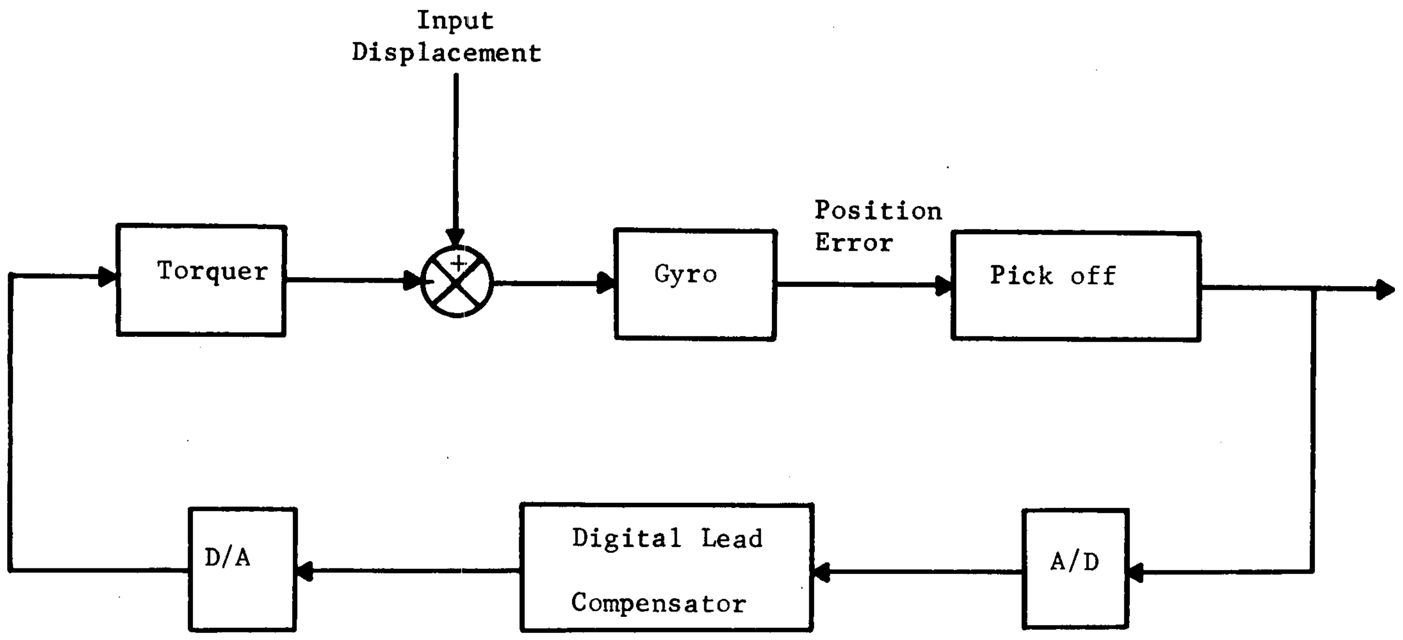


Fig. 13--Error correcting control loop with digital lead compensation.

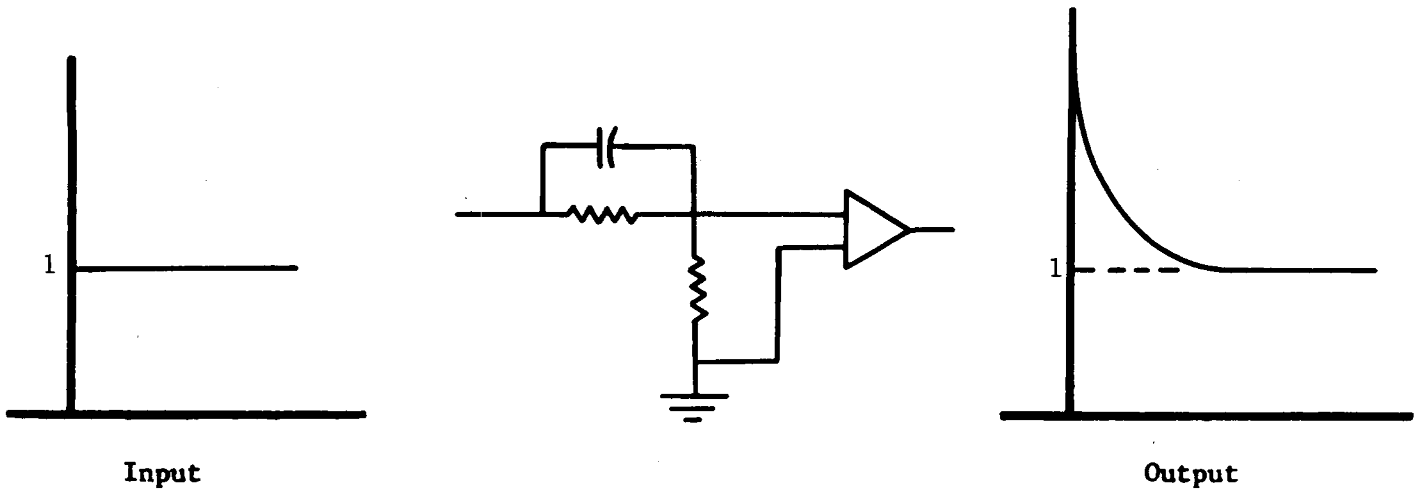


Fig. 14--Analog lead compensation.

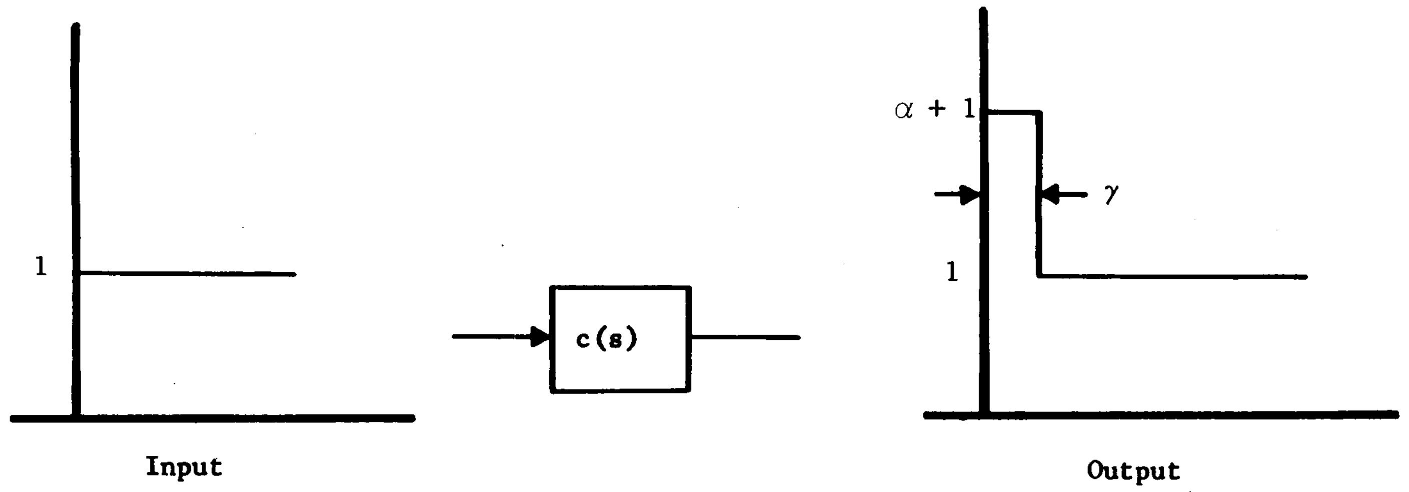


Fig. 15--Digital lead compensation.

The proportional section of the compensator consists of a combinational network which multiplies the input by the appropriate constant. The derivative section of the compensator may be implemented with a bi-directional counter that counts in the desired sequence upon acceptance of an input. The sequence begins at a positive maximum and counts down, if the input is positive, and begins at a negative maximum and counts up toward zero, if the input is negative.

The output sequence for the derivative compensation may also be generated with a sequential machine. A special class of sequential machines, linear sequential machines, are by definition, sequential machines whose next state is a linear function of the present inputs and the previous state. Linear sequential machines may be implemented using only shift registers and modulo-two adders ("exclusive - or" elements), thereby making the linear sequential machine an attractive approach.

It is obvious that the derivative section of the compensator is the most difficult to implement. A block diagram view of the digital compensator is given in Figure 16. Combinational network number one (1) is the proportional section of the compensator, while combinational network number two (2) allows the counter to count in a decay type (non-linear) rather than a linear sequence. However, suppose that the proportionality constant is one, and that the derivative sequence is linearized. This results in a reduction in the size and complexity of the compensator as shown in Figure 17.

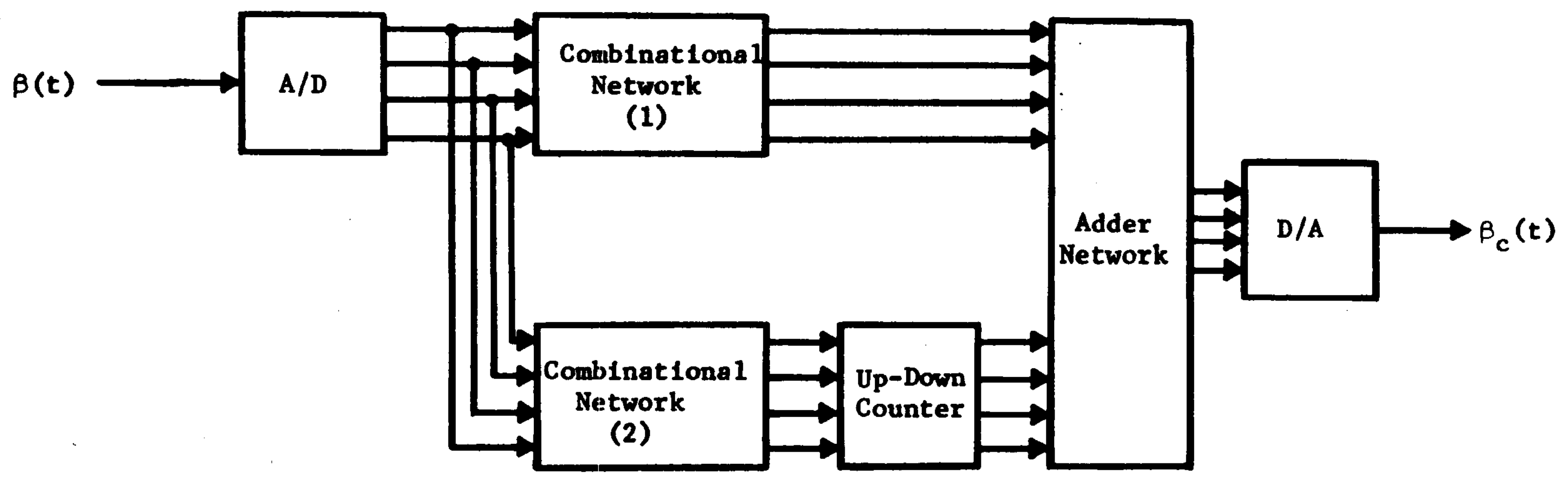


Fig. 16--Block diagram of a digital lead compensator.

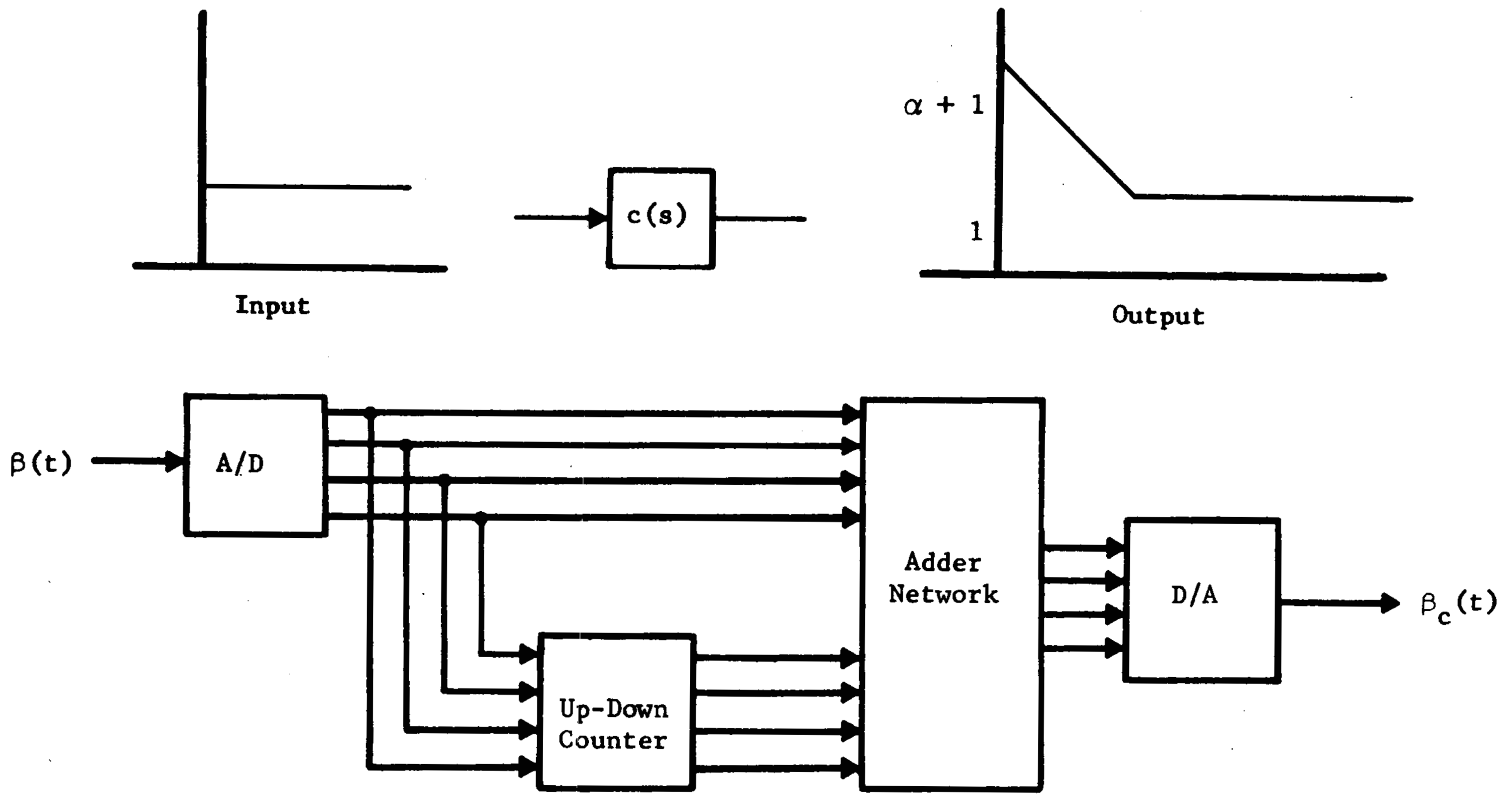


Fig. 17--Block diagram of reduced digital lead compensator.

There are two ways to insure that every input step will be compensated. The first way is to have the input sampling interval greater than the time necessary to generate the derivative count sequence. The second way is to have the sequence generator capable of beginning its sequence upon acceptance of an input, regardless of its position in the previous count sequence. The latter approach was chosen for the compensator of Figure 17. Figure 18 depicts the function of the compensator upon acceptance of an input before completion of the count sequence.

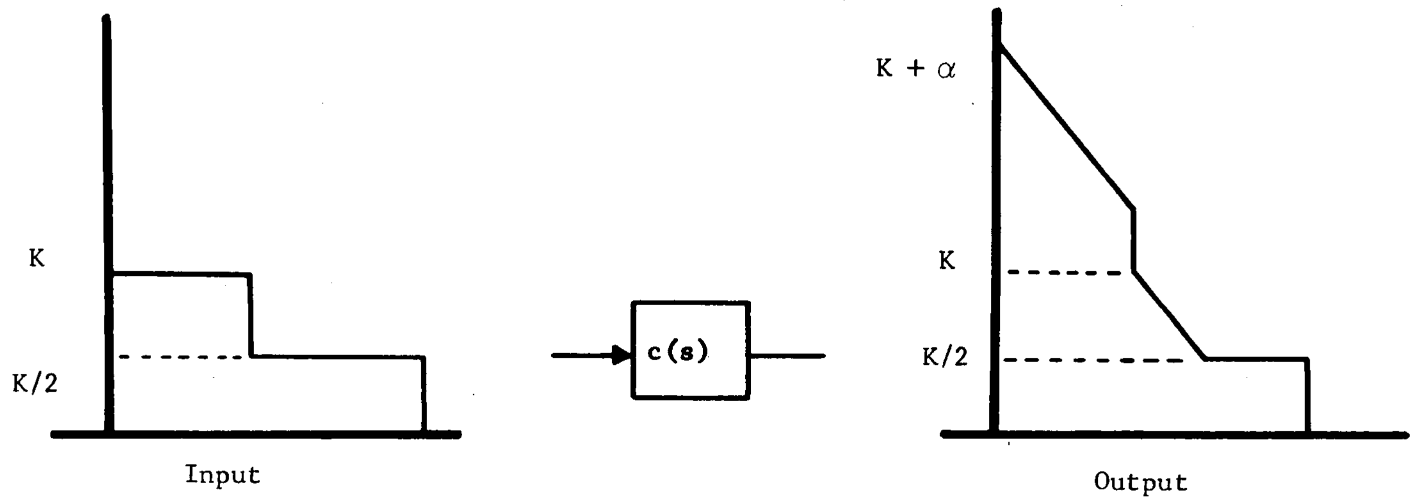


Fig. 18--Function of compensator with example input.

V. CONCLUSIONS

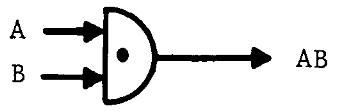
The three synthesis techniques described were derived in a manner suitable for digital implementation and can, theoretically, be used to realize any transfer function that is realizable. However, for systems that demand extreme accuracy, the required logical networks tend to become very large. The most practical of the logical networks is that used in implementing the network equation of Method III, and is shown in Figure 6.

The most feasible method of implementation is that employing a digital computer. These techniques can be extremely useful in time-sharing systems employing on-line digital computers.

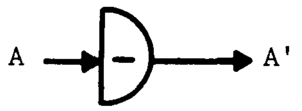
An approach that has not been investigated is the use of a sequential machine to implement the derived network equations. This approach seems very likely to produce good results, since the response at a particular sampling instant is a function of the present input and the output at the previous sampling instant. A special class of sequential machines, linear sequential machines,[9,10] is presently under study and is being considered for implementation of the network equations of Methods I, II, and III.

APPENDIX

ILLUSTRATION OF LOGICAL GATES AND LOGICAL SYMBOLS



AND gate



NOT gate

REFERENCES

1. J. L. Dooley, "Active Compensation Networks Stabilize Guidance Platform," Systems Design, pp. 36-42, April, 1965.

2. J. R. Raggazzini and G. F. Franklin, Sampled-Data Control Systems, New York: McGraw-Hill, 1958, pp. 145-148.

3. D. J. Gawlowicz et al., "Compensating Pulse-data Servos, "Control Engineering, pp. 61-65, June, 1965.

4. J. B. Kidd et al., "Transfer Function Synthesis in the Time Domain - an Extension of Levy's Method," IEEE Trans. on Education, pp. 62-67, June-September, 1965.

5. W. H. Chen, The Analysis of Linear Systems, New York: McGraw-Hill, 1963.

6. R. E. Miller, Switching Theory, Combinational Circuits, Vol. 1, New York: John Wiley and Sons, 1965.

7. R. E. Miller, Switching Theory, Sequential Circuits and Machines, Vol. 2, New York: John Wiley and Sons, 1965.

8. J. Millman and H. Taub, Pulse, Digital, and Switching Waveforms, New York: McGraw-Hill, 1965.

9. A. Gill, Introduction To The Theory of Finite-State Machines, New York: McGraw-Hill, 1962, pp. 169-179.

10. W. W. Peterson, Error-Correcting Codes, Cambridge, Mass.: M.I.T. Press, 1961, pp. 107-117.